UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/837,671	04/18/2001	David L. Detlefs	004-5723	9850

22120 7590 12/02/2004

ZAGORIN O'BRIEN & GRAHAM, L.L.P.
7600B N. CAPITAL OF TEXAS HWY.
SUITE 350
AUSTIN, TX 78731

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2127	

DATE MAILED: 12/02/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/837,671	Applicant(s) DETLEFS ET AL.	
	Examiner Lewis A. Bullock, Jr.	Art Unit 2127	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 21 July 2004.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-35 is/are pending in the application.

4a) Of the above claim(s) 22-28 is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-3,7-10,13-19,21 and 29-35 is/are rejected.

7)☒ Claim(s) 4-6,11,12 and 20 is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Election/Restrictions

1. Applicant's election of Group I, claims 1-21 and 29-35 in the reply filed on 7/21/04 is acknowledged. Because applicant did not distinctly and specifically point out the supposed errors in the restriction requirement, the election has been treated as an election without traverse (MPEP § 818.03(a)).

2. This application contains claims 22-28 drawn to an invention nonelected with traverse in Paper No. 7/21/04. A complete reply to the final rejection must include cancellation of nonelected claims or other appropriate action (37 CFR 1.144) See MPEP § 821.01.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1, 2, 7-9, 13-17, 29, 30 and 33-35 are rejected under 35 U.S.C. 102(b) as being anticipated by "Managing Long Linked Lists Using Lock-Free Techniques" by FAROOK.

As to claim 1, FAROOK teaches a method of providing storage reclamation in a multiprocessor computer system (a parallel environment in which each process executes alone on a processor) (pg. 15, Performance of the New Algorithm), the

method comprising: maintaining respective reference counts (counter fields) for shared objects (list nodes) (pg. 9, "Each node in the linked list consists of four fields; a data field, two pointer fields, and a count field...the counter field in each list node is used..."); accessing pointers to the shared objects (list nodes) using lock-free pointer operations (cursor / TryDelete of delete / TryInsert of insert) to coordinate modification of respective reference counts (see figs. 6-9; pg. 9, section 4.2 Linked List Traversal – pg. 13, "Otherwise, the 'prev' node's counter field is decremented (line 14) and the appropriate result status is returned."); free storage associated with a particular one of the shared objects (list nodes) only once the corresponding reference count (counter fields) indicates that the particular shared object is unreferenced (pg. 12, "The code then decrements the counter of node 'prev' (as it is no longer being used) and, if possible releases the deleted node (line 3) to reclaim space."; pg. 9, "A process attempting a node deletion must first verify that its counter field is zero before proceeding.").

As to claim 2, FAROOK teaches the lock-free pointer operations (cursor / TryDelete of delete / TryInsert of insert) ensure that: if a number of pointers referencing the particular shared object (list nodes) is non-zero, then so too is the corresponding reference count (via as the user moves from node to node it increments the counter field of the node its visiting and decrements it when it is leaving) (pg. 9, 2nd paragraph); and if no pointers reference the particular shared object (list nodes), then the corresponding reference count eventually becomes zero (via the reference count is not increment since no node is visiting it) (pg. 9, 2nd paragraph).

As to claim 7, FAROOK teaches the pointer operations include a destroy operation (delete operation) that: decrements a reference count (counter field) of a shared object (list node) identified by a supplied pointer value (key); and frees the identified shared object if the corresponding reference count has reached zero (via release instruction) (see fig. 6 and 7, pgs.11-12).

As to claim 8, FAROOK teaches wherein, prior to the freeing (via release instruction), the destroy operation (delete operation) recursively follows pointers defined in the shared object (list node) if the corresponding reference count (counter field) has reached zero (via TRY_AGAIN / DELETE_AGAIN instructions) (see fig. 6 and 7, pgs.11-12).

As to claim 9, FAROOK teaches employed in access operations (cursor / TryDelete of delete / TryInsert of insert) on a composite shared object (linked list) that includes zero or more of the shared objects (list nodes) (pg. 8-13).

As to claim 13, FAROOK teaches a lock-free implementation of a concurrent shared object (linked list) comprising: plural component shared objects (list nodes) encoded in dynamically-allocated shared storage (shared memory machine) (abstract; pg. 9, "Each node in the linked list consists of four fields; a data field, two pointer fields, and a count field...the counter field in each list node is used..."); and access operations

(cursor / TryDelete of delete / TryInsert of insert) that, prior to attempting creation or replication of a pointer (accessing a particular node) to any of the component shared objects (list node), increment a corresponding reference count (counter field), and upon failure of the attempt (process has failed due based on DCAS or CAS operation) (pg. 14-15), thereafter decrement the corresponding reference count (counter field), the access operations (cursor / delete / insert) decrementing a particular reference count (counter field), except when handling a pointer creation failure, no earlier than upon destruction of a pointer to a corresponding one of the component shared objects (list nodes) (see figs. 6-9; pg. 9, section 4.2 Linked List Traversal – pg. 13, “Otherwise, the ‘prev’ node’s counter field is decremented (line 14) and the appropriate result status is returned.”; pg. 12, “The code then decrements the counter of node ‘prev’ (as it is no longer being used) and, if possible releases the deleted node (line 3) to reclaim space.”; pg. 9, “A process attempting a node deletion must first verify that its counter field is zero before proceeding.”; see also pg. 14-15).

As to claim 14, FAROOK teaches the access operations employ lock-free, reference-count-maintaining pointer operations (via the delete and insert operations use of CAS or DCAS) (See Fig. 6-9, pgs. 11-13).

As to claim 15, FAROOK teaches a lock-free, reference count maintaining destroy operation (delete operation) (pg. 11-12).

As to claim 16, FAROOK teaches each of the access operations (insert / delete) are lock-free (non-blocking) (abstract; pg. 9; pg. 17, Conclusions & Future Work).

As to claim 17, FAROOK teaches the access operations (insert / delete) employ either a compare-and-swap primitive or a double compare-and-swap primitive (via the delete and insert operations use of CAS or DCAS) (See Fig. 6-9, pgs. 11-13).

As to claim 29, FAROOK teaches a computer program product (computer instructions) encoded in at least one computer readable medium comprising (shared memory machine) (abstract; pg. 9, "Each node in the linked list consists of four fields; a data field, two pointer fields, and a count field…the counter field in each list node is used…"): a representation of a shared object (linked-list) that is instantiable as zero or more component objects (list nodes) in dynamically allocated shared storage (shared memory machine) of a multiprocessor (a parallel environment in which each process executes alone on a processor) (pg. 15, Performance of the New Algorithm); at least one instruction sequence executable by respective processors of the multiprocessor, the at least one instruction sequence implementing at least one access operation (cursor / insert / delete) on the shared object (linked-list) and employing one or more lock-free pointer operations (increment / decrement counter in access operations) to maintain reference counts (counter fields) for one or more accessed component objects (list nodes) thereof (see figs. 6-9; pg. 9, section 4.2 Linked List Traversal – pg. 13, "Otherwise, the 'prev' node's counter field is decremented (line 14) and the appropriate

result status is returned."; pg. 12, "The code then decrements the counter of node 'prev' (as it is no longer being used) and, if possible releases the deleted node (line 3) to reclaim space."; pg. 9, "A process attempting a node deletion must first verify that its counter field is zero before proceeding."; see also pg. 14-15); and the at least one instruction sequence further implementing explicit reclamation of the component objects (list nodes), thereby freeing storage associated with a particular one of the component objects (list nodes) only once the corresponding reference count (counter fields) indicates that the particular component object is unreferenced (via the instruction that if target counter or prev counter is equal to 0 then to release that node) (see fig. 7, pg. 12).

As to claim 30, FAROOK teaches the zero or more component objects (list nodes) of the shared object (linked list) are organized as a linked-list; and wherein the at least one access operation (cursor / insert / delete) supports concurrent access to the linked-list (non-blocking concurrent access) (pg. 9; abstract; pg. 14-15, "We begin by noting that non-adjacent updates do not affect one another since, synchronization is localized (i.e. fine granularity concurrency control).").

As to claim 33, FAROOK teaches the computer readable medium is electronic storage medium (shared memory machine) (abstract).

As to claim 34, FAROOK teaches an apparatus (system) comprising: plural processors (a parallel environment in which each process executes alone on a processor) (pg. 15, Performance of the New Algorithm); a store addressable by the plural processors (shared memory storing the linked list); one or more shared pointer variables (head / tail pointers) accessible by each of the plural processors for referencing a shared object (linked list) encoded in the store; means for coordinating competing access to the shared object (linked list / list node) using one or more reference counts (counter fields) and pointer manipulations (via instructions manipulating the next or prev pointer values in the cursor function / delete function / insert function) that employ one or more lock-free pointer operations (cursor / delete / insert) to ensure that if the number of pointers to the shared object (linked list / list node) is non-zero (node is being referenced / visited) particular , then so too is the corresponding reference count (via each process increments the counter of a node it is about to traverse) and further that if no pointers reference the shared object (process is leaving node), then the corresponding reference count (counter field) eventually becomes zero (via each process decrements the counter of a node it is leaving) (pgs 8-13).

As to claim 35, FAROOK teaches means for freeing the shared object (list node) only once the corresponding reference count (counter field) indicates that the shared object is unreferenced (via the instruction that if target counter or prev counter is equal to 0 then to release that node) (see fig. 7, pg. 12).

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 3 and 21 is rejected under 35 U.S.C. 103(a) as being unpatentable over "Managing Long Linked Lists Using Lock-Free Techniques" by FAROOK.

As to claim 21, FAROOK teaches wherein the concurrent shared object is a linked-list (pg. 8); and wherein the access operations (insert / delete) are performed using a synchronization primitive (DCAS / CAS) to mediate concurrent execution thereof (pg. 8-15). However, FAROOK does not teach the linked list is a doubly linked list.

Official Notice is taken in that it is well known in the art at the time of the invention that a doubly linked list is a version of a linked list wherein each node has a pointer to the next and previous nodes. Therefore, it would be obvious to one skilled in the art that the linked list of FAROOK can be implemented as a doubly linked list and allow concurrent execution to itself by the cited access operations.

As to claim 3, FAROOK teaches that as concurrent processes access a shared object, i.e. visit a shared node of the shared object, they increment that nodes reference count (abstract, pg. 8-15). It is obvious to one skilled in the art that since accesses to

the shared object are concurrently that when both processes access the object at the same time neither would have the true reference count, i.e. if the reference count of a shared object is one and concurrent access is made by concurrently accessing processes each process would increment to two wherein the correct amount of pointers is three.

7. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over "Managing Long Linked Lists Using Lock Free Techniques" by FAROOK in view of "Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms" by MICHAEL.

As to claim 10, FAROOK teaches the composite shared object is a linked list and the shared objects are nodes of the linked list and wherein the access operations implement push (insert) and pop (delete) accesses to opposing ends of the linked list (pg. 8-13). However, FAROOK does not teach the linked list is a doubled ended queue.

MICHAEL teaches a queue implemented as a linked list having a head and tail pointers wherein nodes are pushed (enqueue operation) after the last node in the linked list and pop (dequeue operation) from the beginning of the linked list (pg. 3-5, Sections Algorithms (2) and Correctness (3)). It would be obvious that based on the combination that FAROOK's linked list is a queue capable of being manipulated as disclosed in MICHAEL. Therefore, it would be obvious to one skilled in the art to combine the teachings of FAROOK with the teachings of MICHAEL in order to facilitate

access operations of a non-blocking concurrent queue to proceed concurrently (abstract).

8. Claims 18 and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Managing Long Linked Lists Using Lock-Free Techniques" by FAROOK in view of "Transactional Memory: Architectural Support for Lock-Free Data Structures" by HERLIHY.

As to claim 18, FAROOK substantially discloses the invention above. However, FAROOK does not mention the access operations employ emulations of the compare-and-swap operation.

HERLIHY teaches the access operations (enqueue / dequeue) employ emulations of either or both of the compare-and-swap and double-compare-and-swap operations (via transaction memory) (pg. 8, section 5.3, Doubly-Linked List Benchmark; abstract). Therefore, it would be obvious to one skilled in the art to combine the teachings of FAROOK with the teachings of HERLIHY in order to facilitate efficient lock-free synchronization based on mutual exclusion (abstract).

As to claim 19,HERLIHY teaches wherein the emulation is based on transactional memory (pg. 8, section 5.3, Doubly-Linked List Benchmark; abstract).

9. Claims 31 and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Managing Long Linked Lists Using Lock-Free Techniques" by FAROOK in view of

"Garbage Collection: Algorithms for Automatic Dynamic Memory Management" by JONES.

As to claim 31, FAROOK teaches component objects (list nodes) having a reference count (counter field) (pg. 8). However, FAROOK does not teach using a mutator to provide explicit reclaimation.

JONES teaches partially implementing a mutator (mutator) that provides explicit reclamation of the dynamically allocated shared storage (concurrent reference counter) (via notifying the collector) (pg. 200-201). Therefore, it would be obvious to combine the teachings of FAROOK with the teachings of JONES in order to efficiently manipulate reference counters in a concurrent environment (pg. .200).

As to claim 32, JONES teaches partially implementing a garbage collector (garbage collector) that reclaims shared storage dynamically-allocated from a mutator (mutator) (pg. 200-201).

Allowable Subject Matter

10. Claims 4-6, 11, 12, and 20 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

11. The following is a statement of reasons for the indication of allowable subject matter: The cited claims contain allowable subject matter for at least the following reasoning: All of the claims detail a plurality of lock-free pointer operations, i.e a load

operation, a store operation, a copy operation, and a push access operation that are not taught by any of the prior art of record. Each of the cited operations perform a plurality of compare-and-swap and double-compare-and swap primitives that affect not only the reference count but also the pre-load value of the local pointer value, the pre-store value of the shared pointer variable, the pre-copy value of the local pointer value, or the in pre-splice pointer values. By performing DCAS and CAS operations as disclosed in the claims, the invention allows for the separation of the updates of reference counts from the updates of the pointers themselves (pg. 7, par. 1021). All of the prior art of record detail common practices of allowing the pointers to record the exact number of pointers to an object. FAROOK teaches this common practice in that each insert or delete operation manipulates the counter for the accessed node however this manipulation is not performed by the DCAS or CAS operations with the access operations. Therefore, the claims are allowable over the prior art of record.

Response to Arguments

Applicant's arguments filed 7/21/04 have been fully considered but they are not persuasive. Applicant argued that Farook's reference count only reflects the number of processes currently accessing a shared node of a list and does not reflect the state or structure of the list. The examiner disagrees. First, Applicant's claims does not indicate that the shared object is a list. The claims at best teaches maintaining reference counts for shared objects. Therefore, the shared objects can be nodes of a list as Applicant has argued. Secondly, the claims do not teach that the reference count reflect the state

or structure of the list. The examiner cannot find any limitations in the claims that the reference counts denote the state or structure of the list. In addition, hypothetically even if such a limitation was explicitly claim, since the reference count give an indication of how many processes access the shared object, this would indicate a state of the shared object. Therefore, the limitation would still be met by the teachings of Farook.

Applicant then argues that the Farook suffers from flaws that allows for premature reclamation of a shared object and additional failures while the described invention does not allow for premature reclamation. In response, the examiner would like to point out that the teachings of Farook meet the limitations of the claims as disclosed. Applicant's invention may disallow for premature reclamation, however, how this disallowance is achieved is not denoted in the claims. The cited limitation, Applicant is referring to that teachings this limitation, "freeing storage associated with a particular one of the shared objects only once the corresponding reference count indicates that the particular shared object is unreferenced" is met by the teachings of Farook. Farook states a process attempting a node deletion must first verify that its counter field is zero before proceeding (pg. 9). Therefore, Farook teaches freeing storage associated with a shared object (node) only when the node's reference count indicates that the shared object is unreferenced, i.e. that the counter field of the node is zero. There is no limitation in this step regarding premature reclamation or the handling of situations of premature reclamation. Farook teachings would also meet the similar limitations in the other claims of "lock-free pointer operations (node deletion) to ensure that if the number of pointers to the shared object is non-zero (no process accessing the

node), then so too is the corresponding reference count (counter field) and further that if no pointers reference the shared object, then the corresponding reference count eventually becomes zero" since when process visit the node, it increments the counter field and decrements it when it is leaving the node and in order to delete the node it must verify that its counter field is zero before proceeding (pg. 9). Again, there is no limitation regarding premature reclamation or the handling of premature reclamation in this limitation. One skilled in the art would not interpret the freeing of storage based on a reference count and coordinating of competing access by ensure that the reference count is zero or becomes zero as limitations that would ensure that a shared object is not premature reclaimed. In object oriented technology, objects are typically reclaimed after their reference counter is indicated as zero.

Applicant argued that the unreliability of Farook's reference counters arises from Farook's timing of modifying the reference counters in relation to creating and destroying pointers to shared objects. In particular in regards to claim 13, Farook does not disclose incrementing a reference count prior to creation of a pointer to a shared object and decrementing a reference count upon destruction of a pointer to a corresponding shared object. The examiner disagrees. As detailed above, Farook teaches whenever a process is visiting the node, it increments the counter field and decrements it only when it is leaving. Therefore, as processes access a shared object via a pointer it increments a reference count and as it stops access a shared object it decrements the pointers. It is inherent within the teachings of Farook that a pointer must be created in order to access a shared object. Farook also teaches that shared

objects are concurrently accessed. Therefore, there exist multiple pointers to a shared object wherein as processes traverse shared objects or perform operations on shared objects, the reference count for that object is increment and decremented as the process access or stop accessing that particular shared object.

Applicant argues that Farook does not disclose or suggest a destroy operation that decrements a reference count of a shared object identified by a supplied pointer value and frees the identified shared object if the corresponding reference count has reached zero because the TryDelete decrements a counter of a node that precedes the node being deleted. The examiner disagrees. Farook teaches that whenever a process is visiting the node, it increments the counter field and decrements it only when it is leaving. A process attempting a node deletion must first verify that its counter field is zero before proceeding (pg. 9). The delete operation of Farook repeatedly tries to delete a particular node. Therefore, by performing the delete operation a process must have access to the previous node as well as the current node. Therefore, a process first increments the target node in order to access the node to make sure it's the correct node, marks the node for deletion by manipulating the pointer values and then decrements the reference count of the node since it has finished accessing the node. The operation also handles that when the node has a reference count of zero that the node is freed. Therefore, the teachings of Farook meet the limitations of the claim as disclosed.

Applicant then argues that the TryDelete operation does not recursively follow pointers of a target node. The examiner disagrees. Part of the TryDelete operation is

to repeatly try to assign the next node after the target node to the previous node until this operation is a success. Therefore, the operation must recursively follow the pointer of a target node each time it tries to assign the next node to the previous node until it succeeds.

Applicant argues that Farook does not disclose or suggest the operations recited in claim 15 in particular the other operations except the lock-free reference count maintaining destroy operation as disclosed by the Examiner. In response, the Examiner would like to point out that the claim language states that "the access operations include one or more of". Therefore, the implementation, broadly defined, has only one operation, i.e. the lock-free reference count maintaining destroy operation as disclosed by the Examiner.

In regards to the arguments to claims 3 and 19, the examiner responds by stating that Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

Conclusion

3. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within

TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is(571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 am - 5:00 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

L a Bullock Jr

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER

November 24, 2004